A-UR 77-2583

TITLE: HELPER: A SUPERINTELLIGENT TERMINAL BASED ON RT-11

CONF-771244-3

AUTHOR(S): DAVID E. SCHULTZ, C-1

SUBMITTED TO: DECUS MEETING
SAN DIEGO, CALIFORNIA
November 28 - DECEMBER 1 , 1977

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

Form No. 836
St. No. 2629
1/75

UNITED STATES
ENERGY RESEARCH AND
DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

HELPER: A SUPERINTELLIGENT TERMINAL BASED ON RT-11*

David E. Schultz
Los Alamos Scientific Laboratory
Los Alamos, New Mexico

ABSTRACT

The HELPER minicomputer hardware and software capabilities are thoroughly explained and critically reviewed. HELPER is a cost-effective assistant to an interactive host system. The recommendations are to assign specific types of tasks to HELPER and to study the resulting effects on users. Improvements to the present HELPER hardware are also suggested.

INTRODUCTION

A little over a year was spent exploring the possibilities of using a minicomputer as an improved terminal linked to the Central Computing Facility (CCF) at Los Alamos Scientific Laboratory (LASL). From the onset of the HELPER project, there were three basic questions to answer:

- Can a minicomputer mimic a standard terminal accurately enough to use existing (host) CCF systems without software or hardware modifications?
- Can a Higher Level Job Control Language (HLJCL) be defined which improves the user interface?
- Can a minicomputer that is used as an intermediary between the host system and the user increase the computing power and effectiveness of the interactive user?

As a result of the HELPER investigation, it can be stated that a minicomputer can be used to mimic a standard terminal. Communication between the HELPER PDP-11 and the CCF has been going on for two years with no detrimental effects on the host. Also, dial-up connections have been successfully made between HELPER and NOS (CDC 6000 system), the CTR PDP-10, and several 370s without hardware or software modifications.

Improving the user interface has been a difficult function to analyze. A well-designed HLJCL for the HELPER configuration would probably enhance the interface, but this aspect was not studied in detail in the final phases of the project. Suggested modifications, discussed later in the report, may improve the user interface.

The most important finding of the HELPER project is that the minicomputer, as a powerful terminal connected to the computational abilities of the host, provides a very effective system for the user. This brief review of the project may provide a basis for understanding this conclusion. For a more complete description of the various phases of the project see Ref. 1.

*Research supported by the U.S. Energy Research and Development Administration.

HELPER'S EVOLUTION

Initial Considerations

The original orientation of the HELPER project centered on a dedicated process to support an HLJCL, an improved interactive-user interface and a workable connection to the CCF.

Initial work on the HLJCL indicated that some functions could be performed more logically, concisely, effectively, etc. An HLJCL that supports variables with attributes oriented toward strings as the basic (default) data type and powerful testing and selections operators similar to APL seemed to be the right direction to take. However, user cost of learning a new language and problems with user acceptance of a new language led me to believe that more promising avenues should be pursued.

Foreground/Background (F/B) Capability Increases HELPER's Effectiveness

Early work with a preliminary version of HELPER software, which consumed the entire minicomputer while communicating with the CCF, indicated that such an approach was a poor use of resources. Fortunately, the operating system (RT-11) supplied by the manufacturer provides an F/B capability which allows two jobs to share the machine. The background jobs can be any subsystem of the RT-11, i.e., the compiler, or editor or utility packages. The foreground job can be any user-written code.

Work started on utilizing the F/B capabilities of RT-11 in late fall 1975. After several false starts, an effective use of F/B was implemented in May 1976.

The basic software components of RT-11 F/B are compilers (FORTRAN and BASIC), an editor, a macro assembler, a utility package for manipulating files and directories, a linker, various libraries to provide run-time support, a keyboard monitor and a file system. The hardware used includes a 16k PDP-11/10 (HELPER has also run on an 11/04, 11/20, and 11/40); an AED 3100P dual floppy disk (FD) with controller (DEC floppy disks and hard disks have also been used); and a TELERAY CRT terminal (Superbee CRTs, VT 52s, Tektronix 4013s, LA 36s, and TTYs have also served as the system terminal). Refer to Fig. 1. The operating system resides on one of the two floppy disks. The second FD is used for user files. Each FD can hold about 300k 8-bit characters.

The normal mode of operation is to run in the background. The subsystems of the operating system always run in the background. If desired, a foreground job (high priority) can be linked and loaded with background commands to the RT-11. Two single-character commands to the keyboard monitor control which job (the foreground or background) has control of the one system keyboard and display. The net effect is to have the full capability of the PDP-11 RT-11 available in the background while the foreground job performs other functions.

THE HELPER JOB

The HELPER job, which runs in the foreground (HELPER has also run in the background on both SJ and FB), performs several functions which are discussed in the following sections. Refer to Appendix A for a description of how to execute and communicate with the HELPER job.

HELPER As a Switching Mechanism

The basic task of HELPER is to act as a switch. This switching capability accepts characters typed on the keyboard (and directed to the foreground job) and either sends them to the host or acts upon them as function requests. In the other direction, the HELPER program receives characters from the host and saves them on a file and/or displays them if the display is controlled by the foreground job. Figure 2 shows the message transfer characteristics of the HELPER job. This part of the HELPER job mimics a standard terminal. Host software or hardware changes are not required to run this package. In addition, standard DEC equipment can be used for all HELPER hardware.

HELPER As An Improved User Interface

The second part of the HELPER job is an attempt to improve the user interface. The functions implemented include the ability to send up to 150 predefined lines (commands to the worker machine, or data) with two key strokes. An additional function resends the last line typed without the carriage return. This allows corrections to be made to some erroneous lines without retyping the entire line.

Another function defines the host worker system being used. This has two purposes. First, it allows a common character-delete key (RUB OUT) to be used. RUB OUT is translated to the proper character for the specified system. The second purpose is to redefine the 150 predefined keys for the particular host worker system being used. By consistently defining the keys for each host system, the user interface can be simplified. An example could be to define key "A" to mean DISPOSE(FN=PR) for the Network Operating System (NOS) and ALLOUT FN / 1 1 for the Livermore Time Sharing System (LTSS).

An additional capability is provided to anticipate the proper system-dependent prompt when a file transfer is in progress.

HELPER As a Tool to Solve User Problems

The final part of the HELPER job is to try to provide the user at the terminal with leverage to solve his problem. This leverage is provided in the foreground and in the background.

The foreground leverage comes from the augmented keyboard (the 150 "function" keys described previously), consistency between systems in key definition and file manipulating functions. The file manipulating functions allow a user to send a file from the local FD to the host worker machine or to save information coming from the host on the local FD. Control functions to stop sending a file, stop receiving a file, rewind the files or terminate the foreground job are also provided. The file transfer functions provide a powerful lever since they can be going on in the foreground while the full power of RT-11 is available in the background! An additional built-in feature retransmits a message up to 12 times in case the host indicates an error response on a given line.

RT-11 SUBSYSTEMS INCREASE PRODUCTIVITY

The RT-11 system components running in the background give the terminal user the additional leverage of a local character or line editor, FORTRAN compiler, linker, file manipulating utility program and any user-developed FORTRAN program.

An example of one of these subsystems or jobs which can run in the background while the HELPER foreground job is transferring a file between the local FD and the worker machine is called FW. FW stands for FORTRAN WRITER and is a simple example of the kind of leverage the terminal user can build for himself in a familiar language, FORTRAN.

FW is to the problem of writing FORTRAN programs as HELPER is to the problem of interacting with the CCF worker machines. Features of FW are tailored to making the job of writing FORTRAN programs easier. Twenty-four function keys are available to define commonly character strings. In addition, one

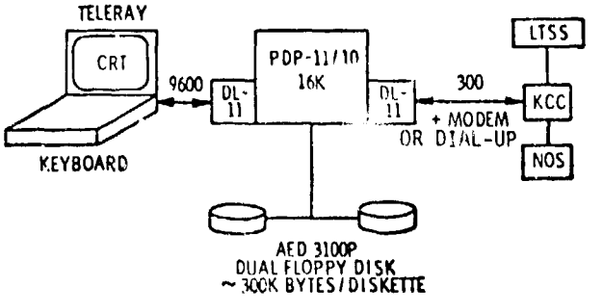

Fig. 1. The HELPER hardware configuration.

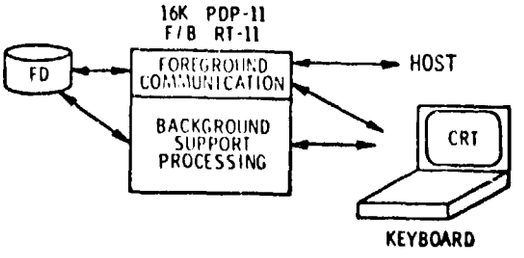

Fig. 2. Foreground communication switches data between host, CRT, and FD.

key is used to keep track of statement-number assignments for FORMAT statements. A set of functions for a simple TAB, increment TAB and decrement TAB is also available to assist in writing structured programs with several levels of indentation to improve code readability (see Appendix B).

These features are a primitive attempt at providing tools to make writing (FORTRAN) programs slightly easier. Since FW is a FORTRAN code which is basically table-driven it appears to be easy for a user to tailor FW to his style or to create a BW (BASIC WRITER) or CW (COBOL WRITER) or RW (RUN FORTRAN WRITER) with little effort.

HELPER'S ADVANTAGES AND POSSIBILITIES

One reaction to the HELPER approach of improving the user interface and providing leverage to solve program generation, testing, and documentation problems might be that all of these functions could be performed on existing worker machines and/or on a different time-shared system. It is my contention that the available alternative systems are inferior for the following reasons.

1. Response time is poorer and less predictable.
2. System availability is less reliable.
3. Data transmission speeds are slower. The rate between the CRT and the PDP-11 is 9600 bps.
4. Allocation of resources such as permanent file space is much more difficult and frustrating for the user.
5. These systems must be more general-purpose and are not as likely to be easily tailored to a particular user's needs. The HELPER software is written entirely in FORTRAN and runs as a user job. For a complete listing of the HELPER program see Ref. 2.
6. A more limited set of terminal devices is likely to be supported.
7. The cost of software development is higher.

The multifaceted capabilities of the HELPER system, allowing compiler, editor, FORTRAN WRITER, and utility routine executions in one portion of the computer and sophisticated file transfer and control functions in the other portion, indicate that we have an amenable interface for many users.

It is suggested that the HELPER system could relieve the large worker machines of certain kinds of jobs: code editing, inputting and outputting of textual information, limited permanent storage capacity, etc. This freeing of computer time would conceivably aid both the user who is plagued by slow compilation time and the word processor whose jobs are constantly being delayed by number-crunching codes.

AREAS FOR FURTHER STUDY

Problems and Their Solutions

Though the current HELPER can be used to provide substantial aid to a programmer, the project pointed out areas where HELPER could be improved to provide even more aid for a wider range of problems.

Data Rate. The data rate of 300 bps between the HELPER terminal and the Integrated Computer Network (host) appears to be inadequate for some problems.

Using high-speed lines to the host that would accommodate 300 bps for input and 1200 bps for output could be an initial step to increase data rate. Eventually, lines allowing 9600 bps for input and 9600 bps for output could be incorporated. This setup would be better, but some applications may benefit from even higher data rates.

A high-speed path to the host such as the proposed 50k-bps lines could be used. The principal drawback of this pathway is the limited number of lines (five are proposed).

Disk Storage. The user's local FD storage capacity of 300k bytes is inadequate for some applications.

Several manufacturers now produce double-density FD systems which increase storage capacity from 300k bytes to 600k bytes per drive. More than two drives could be attached to the HELPER system, but this arrangement increases the basic system cost by 30-40% for two additional diskette drives.

Eliminating the system disk would make room for 300k bytes (or 600k bytes for a double-density system) of additional user storage. However, down-link loading, even at 9600 bps, would be too slow for many applications.

Another solution to the storage problem could be to attach a RAM3 to HELPER. This device consists of a single FD, a moveable rack which holds up to 32 diskettes and a positioner/diskette changer. Using double density, about 20 000 000 bytes are accessible in less than 10 s.

HELPER's Appearance. The prototype systems are physically unattractive and take up too much space. Some prospective users are overwhelmed by the physical size of the HELPER hardware. There is much unused rack space which makes the HELPER appear larger than it really is.

This problem could be remedied by using the LSI-11 packaged version (PDP-11/03) which takes up considerably less space than the rack-mounted PDP-11/10s do. Low-profile FD systems are also now available.

Inter-HELPER Data Transmission. Communicating data to other HELPER terminal users is only possible through the host or by physically carrying a floppy diskette.

Higher speed lines to the host would allow better communication between HELPER terminals.

Software Support. It would be difficult to maintain and enhance software for a large number of HELPER terminals because each HELPER terminal is completely stand-alone. Each configuration requires a complete software system.

Down-link loading of the RT-11 system would minimize this problem of running multiple-identical operating systems. At the present data rate of 300 bps, this setup is impossible. Even at 9600 bps, the inability to transmit binary data would make the effective data rate too slow to satisfactorily run the operating system.

CONCLUSION

The HELPER terminal system has been evaluated extensively in the past year. A minicomputer with HELPER software is a powerful and cost-effective way to solve a large number of computing problems.

REFERENCES

1. D. E. Schultz, "An Interim Report on the HELPER Project," Los Alamos Scientific Laboratory Report LA-6470-MS (September 1976).

2. D. E. Schultz, "A Final Report on the HELPER Project," Los Alamos Scientific Laboratory Report LA-6680-MS (February 1977).

APPENDIX A

PRELIMINARY DOCUMENTATION FOR F/L HELPER

To load and execute the HELPER in the foreground type:

.FRUN GOOD/N:600 followed by CR

To attach the keyboard (and display) to the foreground job type:

Ctrl F

To return to the background type:

Ctrl B

ALL features of RT-11 are available to the background job.

When the keyboard and display are attached to the foreground job (HELPER), the terminal can be used to indicate that a command to HELPER follows.

In addition, the escape key (ESC) is used to indicate that a command to HELPER follows.

Character after ESC	Meaning
!	Terminate the foreground job
"	Start saving responses on TT.X -RESETS send flag
%	Start sending the file TCCF.DAT -RESETS save flag
#	Clear both send and receive flag
?	Resend last line typed except for CR
A-Z	Send selected predefined string of characters
0-9	Select 1 of 10 sets of predefined strings Redefines A-Z.

******** The following 4 characters select which file the key definitions are loaded from. The file names are:
KEYDEF.LTS for LTSS
KEYDEF.KRO for NOS
KEYDEF.PDP for PDP 10
KEYDEF.DAT for DAT
For instructions on how to build personalized key definitions tailored to your needs, see MAKEY.DOC.

,	Sets system to LTSS and RUBOUT generates CTRL X
-	Sets system to NOS and RUBOUT generates CTRL H
.	Sets system to PDP-10 and RUBOUT generates RUBOUT
/	Sets system to DAT and RUBOUT generates RUBOUT

******** The following 4 characters select which prompt character to wait for after sending a line to the CCF.
See % description for sending a file.

(	Expected prompt character is & for LTSS.
)	Expected prompt character is ? for NOS.
*	Expected prompt character is X ON normally for PDP-10 but also NOS in TAPE mode.
+	Expected prompt character is LF.
:	Display current key definitions.
&	Allow the foreground job to display messages when the background job has the display/keyboard.
'	Prohibit the foreground job from displaying messages when the background job has the display/keyboard.
ESC	Send one ESC to the CCF.

***

One control character (CTRL P) also has special meaning. Since the CRT is local to the PDP-11, BREAK has no significance. To send a BREAK to the CCF type CTRL P.

Note: The default system is LTSS. The default prompt character is &.

***

The HELPER will automatically resend the last line up to 12 times if the system responds with an error message.

***

End HELPER write-up dated 20-MAY-76
Revised 17-NOV-76

APPENDIX B

FORTRAN WRITER CHARACTERISTICS

Escape key definitions for FW
*** ***

KEY	Character string
B	LOGICAL *1
C	COMMON
D	DIMENSION
E	END
G	GO TO
I	INTEGER
K	CONTINUE
L	DO
N	100 (Each time N is used the number is increased by 1)
P	CALL
R	READ(
S	STOP
T	IF(
W	WRITE(
X	RETURN

***

Control functions

ESC * resets the N key counter to 100.

ESC # closes the file KB.DAT.

ESC ! exits the program.

The TAB key spaces to column NCTAB. If the cursor is already past column NCTAB, TAB is a no operation.

ESC + adds one to NCTAB. NCTAB starts at column 7.

ESC - subtracts one from NCTAB.

RUB OUT removes bad characters.

There is no delete line.

***

End FW write-up dated 18-JUN-76
Revised 25-AUG-76